(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 310 864 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
14.05.2003 Bulletin 2003/20

(51) Int Cl.^{7}: G06F 9/38

(21) Application number: 02028088.9

(22) Date of filing: 28.05.1997

(84) Designated Contracting States:
DE FR GB

(30) Priority: 30.05.1996 JP 13621296

(62) Document number(s) of the earlier application(s) in accordance with Art. 76 EPC:
97108644.2 / 0 810 518

(71) Applicant: MATSUSHITA ELECTRIC INDUSTRIAL CO., LTD.
Kadoma-shi, Osaka 571-8501 (JP)

(72) Inventors:
- Yasoshima, Hiroyuki
Menlo Park, Califonia 94025 (US)
- Kabuo, Hideyuki
Izumi-shi, Osaka 594 (US)

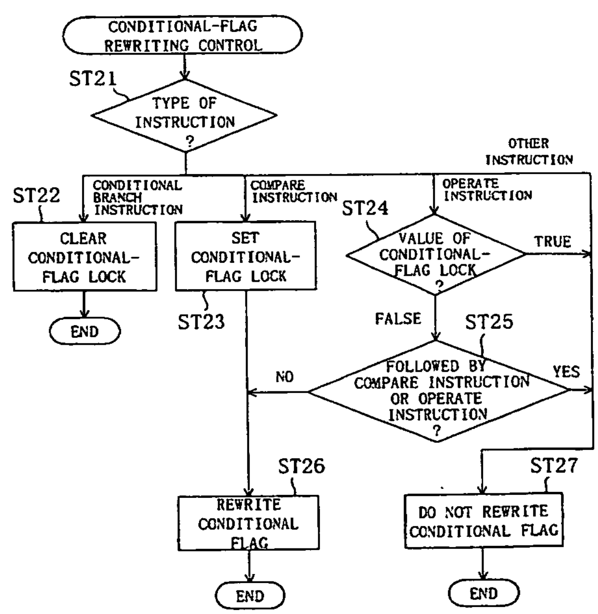
(74) Representative: Grünecker, Kinkeldey, Stockmair & Schwanhäusser Anwaltssozietät
Maximilianstrasse 58
80538 München (DE)

Remarks:
This application was filed on 17 - 12 - 2002 as a divisional application to the application mentioned under INID code 62.

(54) Method and circuit for conditional-flag rewriting control

(57) A method of controlling the rewriting of a conditional flag in a processor comprises a first step of determining, when an instruction involving the rewriting of the conditional flag is executed, whether the rewriting of the conditional flag is enabled or disabled based on the order of instructions in a program. This method further comprises a second step of rewriting, when the rewriting of the conditional flag is enabled, the conditional flag in executing said instruction or not rewriting, when the rewriting of the conditional flag is disabled, the conditional flag in executing said instruction.

Fig. 9



EP 1 310 864 A2

Printed by Jouve, 75001 PARIS (FR)

capacity for storing the program is increased accordingly.

[0012] The problem may arise even when the delayed branch instructions are not consecutively given but they are relatively close in sequence in the program. Although FIGS. 12 and 13 are based on the assumption that the number of delay slots in the processor is 1, a similar problem occurs when the processor has a larger number of delay slots and the spacing between inconsecutive delayed branch instructions is small relative to the number of delay slots in the processor, which complicates the sequence in which the instructions are executed when individual branch conditions for the instructions are satisfied.

[0013] There is another type of processor wherein a control bit for determining whether or not a delayed branch is implemented is provided in the code of a delayed branch instruction (see Japanese Laid-Open Patent Publication HEI 4-127237). If the control bit is 0, the processor does not execute an instruction placed in a delay slot when a branch is enabled. FIG. 14 shows a relationship between the occurrence or nonoccurrence of a branch specified by each of delayed branch instructions and instruction execution sequence when the program shown in FIG. 11 is executed by the processor of this type. In FIG. 14, "****" indicates that the instruction placed in the delay slot is not executed. In this case, if the control bit for the first delayed branch instruction br200 is set to 0, the first delayed branch instruction br200 determines whether or not a branch to the address 200 is implemented and the second delayed branch instruction br400 is executed only when no branch occurs, so that the readability of the program on the assembler level is not impaired.

[0014] However, the provision of the control bit for determining the occurrence or nonoccurrence of a delayed branch in the instruction code increases the bit width of the instruction code by 1 bit, so that the memory capacity for storing the program is increased disadvantageously. In particular, an increased memory capacity forms a fatal drawback to a portable telecommunication device in terms of device size, power consumption, manufacturing cost, and the like. Moreover, the processor requires an additional circuit for controlling delayed branch based on the control bit in the delayed branch instruction code, which is to be provided in the instruction decoder of the processor.

[0015] Furthermore, in the case where the bit width of the instruction code is preliminarily determined by specifications, the provision of even one control bit adds constraints to device design. If the instruction code is composed of 24 bits, e.g., at most only several bits other than the bits required to represent an instruction type, a specified address, and the like can be used for design. As a result, even one control bit may significantly reduce design flexibility.

[0016] A similar problem also occurs in the conventional method of controlling the rewriting of a conditional flag. The provision of a control bit for the rewriting of the conditional flag causes the problem of an increased memory capacity for storing a program, the problem that the instruction decoder requires an additional internal circuit for controlling the rewriting of the conditional flag based on the control bit in an instruction code, and the problem of reduced flexibility with which device design is conducted.

SUMMARY OF THE INVENTION

[0017] The present invention has been achieved to implement delayed branch control in a processor employing a delayed branch method, wherein instruction execution sequence is not complicated and the readability of a program on the assembler level is improved without providing a control bit in an instruction code.

[0018] The present invention also provides a method of controlling the rewriting of a conditional flag without providing a control bit in an instruction code.

[0019] Specifically, the present invention provides a method of controlling a delayed branch operation in a processor employing a delayed branch method. In executing a delayed branch instruction, when a branch has occurred in a specified one or ones of a continuous sequence of cycles immediately before an execute cycle for the delayed branch instruction which are equal in number to delay slots in the processor, the branch specified by the delayed branch instruction is disabled.

[0020] In accordance with the method, even if the branch condition for the delayed branch instruction is satisfied, the branch is disabled when a branch has occurred in the specified cycle previous to the execute cycle for the delayed branch instruction, so that the branch is 'not implemented. Consequently, instruction execution sequence is not complicated even in the case where individual branch conditions are satisfied for delayed branch instructions close to each other relative to the number of delay slots in the processor, which improves the readability of the program on the assembler level.

[0021] The present invention also provides, as a circuit implementing the delayed branch control method, a delayed branch control circuit provided in a processor employing a delayed branch method to control a branch operation, the circuit comprising: a branch-information storing circuit for storing information indicating whether or not a branch has occurred in a specified one or ones of a continuous sequence of cycles immediately before a current execute cycle which are equal in number to the number of delay slots in the processor; and a branch judging circuit for directing, in executing the delayed branch instruction, the processor to implement a branch only when a branch condition for the delayed branch instruction is satisfied and the branch-information storing circuit stores information indicating that no branch has been implemented in the specified cycle or cycles.

[0022] Preferably, the branch-information storing circuit comprises a shift register composed of flip-flops

operation thereof;

FIGS. 6 show still another example of the conditional-flag rewriting control circuit according to the embodiment of the present invention, of which FIG. 6 (a) shows the structure thereof and FIG. 6(b) shows the operation thereof;

FIG. 7 is a block diagram showing the structure of a processor employing a pipeline processing method, which comprises the delayed branch control circuit and conditional-flag rewriting control circuit according to the embodiment of the present invention;

FIG. 8 is a flow chart illustrating the operation of the delayed branch control circuit according to the embodiment of the present invention in the processor shown in FIG. 7;

FIG. 9 is a flow chart illustrating the operation of the conditional-flag rewriting control circuit according to the embodiment of the present invention in the processor shown in FIG. 7;

FIG. 10 shows the flow of pipeline processing during the execution of a branch instruction;

FIG. 11 shows an example of a program on the assembler level when delayed branch instructions are consecutively given;

FIG. 12 shows a relationship between the occurrence or nonoccurrence of a branch in each of the delayed branch instruction and instruction execution sequence when the program shown in FIG. 11 is executed;

FIG. 13 shows the flow of pipeline processing when individual branch conditions are satisfied for the consecutive delayed branch instructions during the execution of the program shown in FIG. 11; and

FIG. 14 shows a relationship between the occurrence or nonoccurrence of a branch in each of the delayed branch instructions and instruction execution sequence when the program shown in FIG. 11 is executed by a type of processor executing an instruction code provided with a delayed branch control bit.

DETAILED DESCRIPTION OF THE INVENTION

[0035] A description will be given first to the principle of delayed branch control according to the present invention.

[0036] As described previously in "BACKGROUND OF THE INVENTION," a processor employing a delayed branch method performs such a complicated operation as to seriously impair the readability of a program on the assembler level only when individual branch conditions for consecutive delayed branch instructions are satisfied. If the branch conditions are unsatisfied, the operation caused by the delayed branch instructions is the same as caused by a NOP (Non-Operate) instruction, which is comparable to substantially no execution.

[0037] When the individual branch conditions for the delayed branch instructions are satisfied, therefore, the operation of seriously impairing the readability of the program on the assembler level can be prevented under such control that the branch condition for the delayed branch instruction placed in a delay slot is constantly unsatisfied.

[0038] FIG. 1 shows the structure of a delayed branch control circuit according to an embodiment of the present invention, which has been implemented based on the principle. In the drawing are shown: a shift register 11 as a branch-information storing circuit; an AND gate 12 as a branch judging circuit outputting a branch indicate signal SI; an inverter 13 for inverting the branch indicate signal SI outputted from the AND gate 12 and inputting the obtained signal to the shift register 11.

[0039] The shift register 11 consists of flip-flops equal in number to delay slots in a processor. In every cycle or every time an instruction is executed by the processor, the shift register 11 receives the branch indicate signal SI inverted by the inverter 13, while shifting the signal held thereby. In short, the shift register 11 stores information indicating the occurrence or nonoccurrence of a branch in each of a continuous sequence of cycles immediately before the current execute cycle which are equal in number to the delay slots in the processor and inputs the stored information as a branch-information store signal SR to the AND gate 12.

[0040] The AND gate 12 outputs the branch indicate signal SI only when it receives a branch execute signal SEa because of the satisfied branch condition for the branch instruction and the branch-information store signal SR received from the shift register 11 indicates that no branch has occurred previously.

[0041] Thus, the delayed branch control circuit shown in FIG. 1 stores, in the shift register 11, information indicating a previous branch caused or not caused by a branch instruction in each of the continuous sequence of cycles immediately before the current execute cycle which are equal in number to the delay slots in the processor and unconditionally disables a branch as long as the shift register 11 stores the occurrence of a previous branch.

[0042] FIG. 2 illustrates the operation of the delayed branch control circuit shown in FIG. 1 when the program shown in FIG. 11 is executed by pipeline processing. In FIG. 2, the number of delay slots in the processor is assumed to be 1. As shown in FIG. 2, when a branch condition is satisfied with the delay branch control circuit shown in FIG. 1 in operation, a branch is enabled if branch information stored in the shift register 11 is false, while a branch is disabled if branch information stored in the shift register 11 is true. Accordingly, when the delayed branch instructions are given consecutively and a branch has been caused by the first conditional branch instruction br200, the second conditional branch instruction br400 is constantly unsatisfied.

[0043] Although the shift register 11 consisting of flip-flops equal in number to delay slots is provided in the

erate signal SDd indicating that the operate instruction is in the decode stage and outputs a signal disabling the rewriting of the conditional flag specified by the operate instruction when the compare signal SDc or operate signal SDd is true, i.e., the compare signal SDc or operate signal SDd is in the decode stage. An AND gate 25 receives the operate execute signal SEd indicating that the operate instruction is in an execute stage and produces a true output signal only when the operate execute signal SEd is true, i.e., the operate instruction is in the execute stage and the rewriting of the conditional flag is not disabled by an output signal from the NOR gate 24. An OR gate 26 outputs the conditional-flag rewrite signal SF when an output signal from the AND gate 25 is true. As a result, the conditional-flag rewrite signal SF is outputted only when the instruction to be subsequently executed is not for rewriting the conditional flag in the execute cycle for the operate instruction. The OR gate 26 constantly outputs the conditional-flag rewrite signal SF when the compare execute signal SEc indicating that the compare instruction is in the execute stage is true.

[0054] As a result, even when the operate instruction (AD1) is in the execute stage, if the operate instruction (AD2) is in the decode stage, the conditional flag can not be rewritten, as shown in FIG. 5(b).

[0055] If the subsequent instruction is a conditional branch instruction in the execute cycle for the operate instruction, the rewriting of the conditional flag is obviously valid. In the second method, therefore, the rewriting of the conditional flag is enabled only when the instruction subsequent to the operate instruction is the conditional branch instruction.

[0056] FIGS. 6 show a conditional-flag rewriting control circuit for implementing the second method according to the embodiment of the present invention, of which FIG. 6(a) illustrates the structure thereof and FIG. 6(b) illustrates the operation thereof. In FIG. 6(a), an AND gate 27 receives a conditional branch signal SDb indicating that the conditional branch instruction is in the decode stage and an operate execute signal SEd indicating that the operate instruction is in the execute stage and produces a true output signal when each of the conditional branch signal SDb and the operate execute signal SEd is true, i.e., when the operate instruction is in the execute stage and the conditional branch instruction is in the decode stage. An OR gate 28 outputs the conditional-flag rewrite signal SF when an output signal from the AND gate 27 is true. As a result, the conditional-flag rewrite signal SF is outputted in the execute cycle for the operate instruction only when the instruction to be subsequently executed is a conditional branch instruction. On the other hand, the OR gate 28 constantly outputs the conditional-flag rewrite signal SF when the compare execute signal SEc indicating that the compare instruction is in the execute stage is true.

[0057] As a result, the conditional flag is rewritten when the operate instruction (AD1) is in the execute stage and the conditional branch instruction (BR) is in the decode stage and the conditional flag is not rewritten when the operate instruction (AD2) is in the execute stage and the conditional branch instruction is not in the decode stage, as shown in FIG. 6(b).

[0058] When the pipeline has an increased number of stages, it is possible to recognize not only the type of the instruction to be subsequently executed but also the types of instructions to be executed thereafter. Hence, it will be appreciated that the rewriting of the conditional flag can be controlled based on the information in accordance with the first or second method.

[0059] FIG. 7 is a block diagram showing the structure of a processor performing pipeline processing, which comprises the delayed branch control circuit and conditional-flag rewriting control circuit according to the embodiment of the present invention. In FIG. 7, the portions other than those associated with delayed branch control and conditional-flag rewriting control shown in FIG. 7 are shown in a simplified manner.

[0060] In FIG. 7, an instruction memory 101 is a memory for storing an instruction code, which outputs an instruction addressed by a program counter (PC) 102 to an instruction decoder 110. The instruction decoder 110 comprises: a branch instruction decoder 111, a compare instruction decoder 112; and an operate instruction decoder 113. The branch instruction decoder 111 selectively decodes a branch instruction among instructions inputted from the instruction memory 101 and outputs the branch signal SDb, while judging whether or not a branch is enabled by referencing the conditional flag register 103 and outputting a branch enable signal SDa when a branch is enabled. The compare instruction decoder 112 selectively decodes a compare instruction among instructions inputted from the instruction memory 101 and outputs the compare signal SDc. The operate instruction decoder 113 decodes an operate instruction among instructions inputted from the instruction memory 101 and outputs the operate signal SDd. The branch enable signal SDa, branch signal SDb, compare signal SDc, and operate signal SDd are timed by a delay circuit (FF) 104 with the branch enable execute signal SEa, branch execute signal SEd, compare execute signal SEc, and operate execute signal SDd, respectively, which are timing signals in the execute stage.

[0061] A delayed branch control circuit 120 has the same structure as the delayed branch control circuit shown in FIG. 1. The delayed branch control circuit 120 consists of a branch-information storing circuit 121 composed of a single flip-flop, an inverting gate 122, and an AND gate 123 and outputs as a branch enable execute signal SEa and an AND signal between the branch enable execute signal SEa and an output signal from the branch-information storing circuit 121. A selector 105 selects an input signal to a PC102 responsive to the branch indicate signal SI. The branch indicate signal SI is held by the branch-information storing circuit 121.

[0062] A conditional-flag-rewriting control circuit 130

compare instruction, e.g., the conditional-flag lock register 131 is set by the compare execute signal SEc and the conditional-flag rewrite signal SF remains false until the conditional-flag lock register 131 is cleared by a subsequent instruction specifying a reference to the conditional flag such as a conditional branch instruction, so that the operation of the conditional-flag generator 141 and the updating of the conditional flag register 103 is halted.

[0070] When operate instructions are given consecutively, the conditional-flag rewrite signal SF becomes false because the AND gate 133 implements the AND operation on the inversion signal of the operate signal SDd and the operate execute signal SEd, so that the operation of the conditional-flag generator 141 and the updating of the conditional flag register 103 is halted.

[0071] As described above, under delayed branch control according to the present embodiment, the program on the assembler level can retain its readability even when delayed branch instructions are given consecutively. Moreover, a circuit implementing the delayed branch control method according to the present embodiment can be implemented by a simple structure. For example, when the number of delay slots in the processor is 1, the circuit can be implemented by a small-scale circuit composed of a single flip-flop with an inverting output and a single AND gate.

[0072] On the other hand, under conditional-flag rewrite control according to the present invention, the rewriting of the conditional flag can efficiently be controlled even when the rewrite control bit for the conditional flag is not provided in the instruction code. This facilitates programming on the assembler level and production of a compiler as well as suppresses power consumed by the operation of the conditional-flag generator and the updating of the conditional flag register in the operating unit. Although operate instructions contained in a program normally accounts for 80% of the entire operation cycle, while branch instructions accounts for 20% thereof, the cycle for performing the operation of the conditional flag generator and the updating of the conditional flag register can be reduced from 80% to 20%.

Claims

1. A method of controlling the rewriting of a conditional flag in a processor, said method comprising:

    a first step of determining, when an instruction involving the rewriting of the conditional flag is executed, whether the rewriting of the conditional flag is enabled or disabled based on the order of instructions in a program; and

    a second step of rewriting, when the rewriting of the conditional flag is enabled, the conditional flag in executing said instruction or not rewriting, when the rewriting of the conditional flag is disabled, the conditional flag in executing said instruction.

2. A conditional-flag rewriting control method according to claim 1, wherein said first step comprises the step of disabling, when an operate instruction is executed, the rewriting of the conditional flag during a period between the execution of a compare instruction and the execution of a conditional branch instruction.

3. A conditional-flag rewriting control method according to claim 1 or 2, wherein said first step comprises the step of disabling, when an operate instruction is executed, the rewriting of a conditional flag if an instruction to be subsequently executed is an instruction specifying the rewriting of the conditional flag.

4. A conditional-flag rewriting control method according to any of claims 1 to 3, wherein said first step comprises
the step of enabling, when an operate instruction is executed, the rewriting of a conditional flag only if an instruction to be subsequently executed is a conditional branch instruction.

5. A conditional-flag rewriting control circuit provided in a processor, said circuit determining whether the processor enables or disables the rewriting of the conditional flag based on the order of instructions in a program.

6. A conditional-flag rewriting control circuit according to claim 5, comprising:

    a conditional-flag lock circuit for setting an output signal therefrom when a compare instruction has been executed and clearing the output signal when a conditional branch instruction has been executed; and

    a rewriting judging circuit for disabling the rewriting of the conditional flag when the output signal from said conditional-flag lock circuit is set in executing an operate instruction.

7. A conditional-flag rewriting control circuit according to claim 5 or 6, wherein said processor decodes and executes an instruction by pipeline processing, said conditional-flag rewriting control circuit comprising a logic circuit for disabling the rewriting of the conditional flag when an instruction in an execute stage is an operate instruction and an instruction in a decode stage specifies the rewriting of the conditional flag.

8. A conditional-flag rewriting control circuit according

Fig. 1

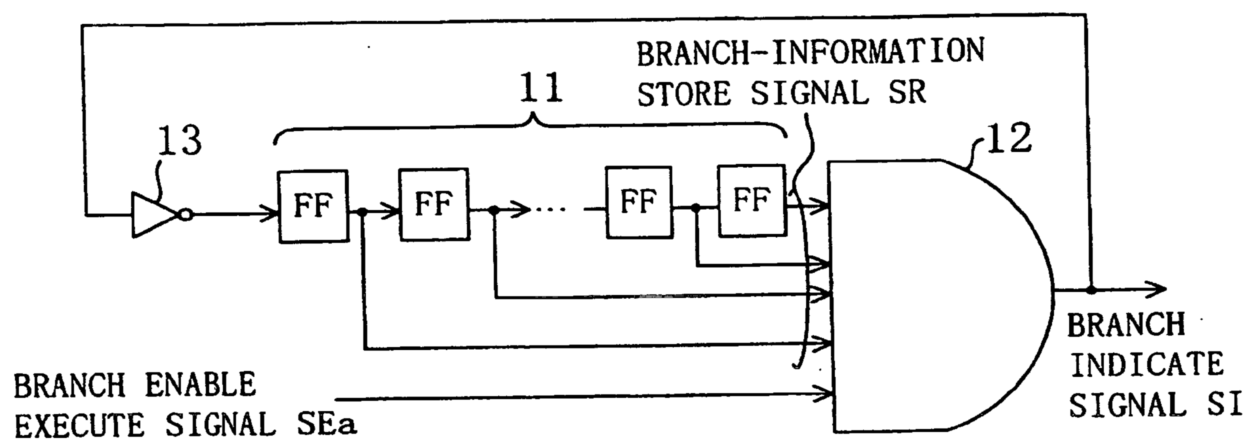

BRANCH-INFORMATION
STORE SIGNAL SR
11
13
12
FF
FF
FF
FF
BRANCH
INDICATE
SIGNAL SI
BRANCH ENABLE
EXECUTE SIGNAL SEa

Fig. 2

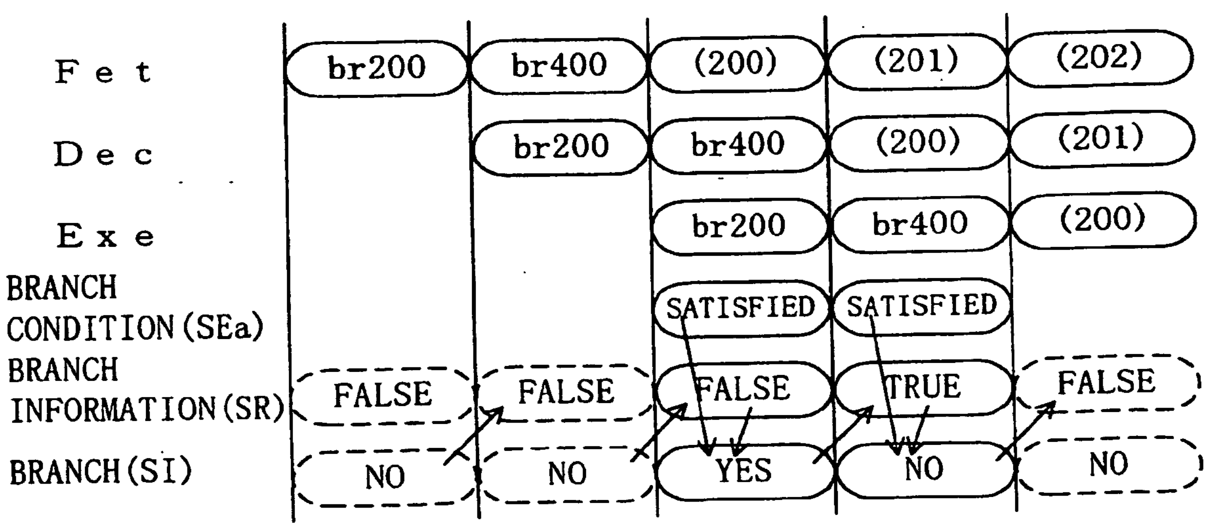

Fet
Dec
Exe
BRANCH
CONDITION (SEa)
BRANCH
INFORMATION (SR)
BRANCH (SI)
br200
br400
(200)
(201)
(202)
br200
br400
(200)
(201)
br200
br400
(200)
SATISFIED
SATISFIED
FALSE
FALSE
FALSE
TRUE
FALSE
NO
NO
YES
NO
NO

Fig. 4(a)

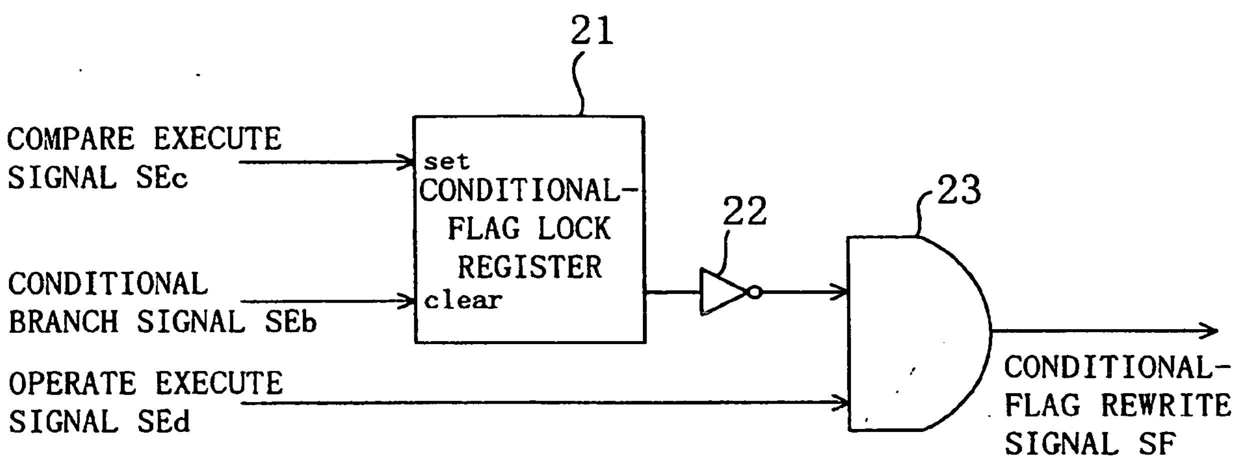


Fig. 4(b)

INSTRUCTION	REWRITING OF CONDITIONAL FLAG SF	CONDITIONAL-FLAG LOCK REGISTER 21
CMP	○	SET
ADD	× LOCKED	——
ADD	× LOCKED	——
BR	——	CLEAR
ADD	○	——
CMP	○	SET
	⋮	⋮

Fig. 6(a)

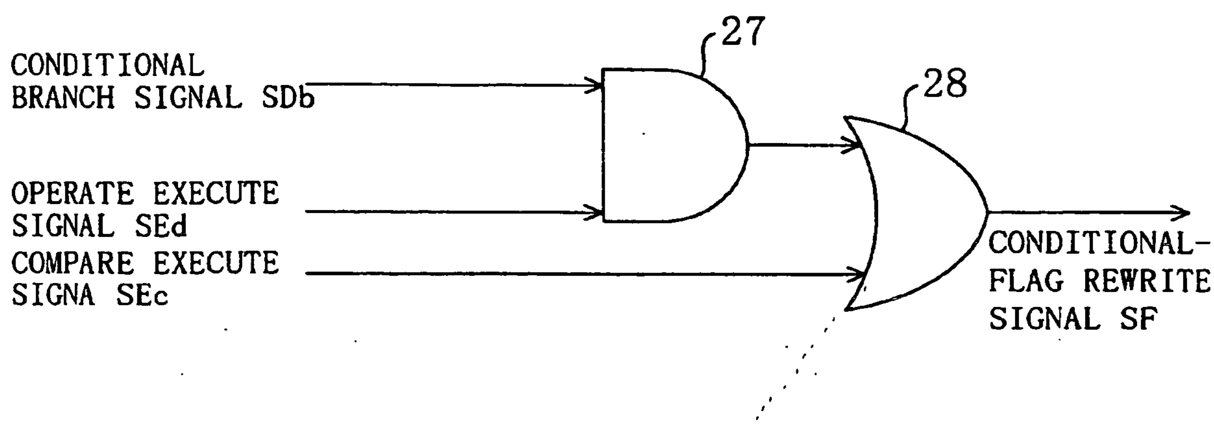

27
28
CONDITIONAL
BRANCH SIGNAL SDb
OPERATE EXECUTE
SIGNAL SEd
COMPARE EXECUTE
SIGNA SEc
CONDITIONAL-
FLAG REWRITE
SIGNAL SF

Fig. 6(b)

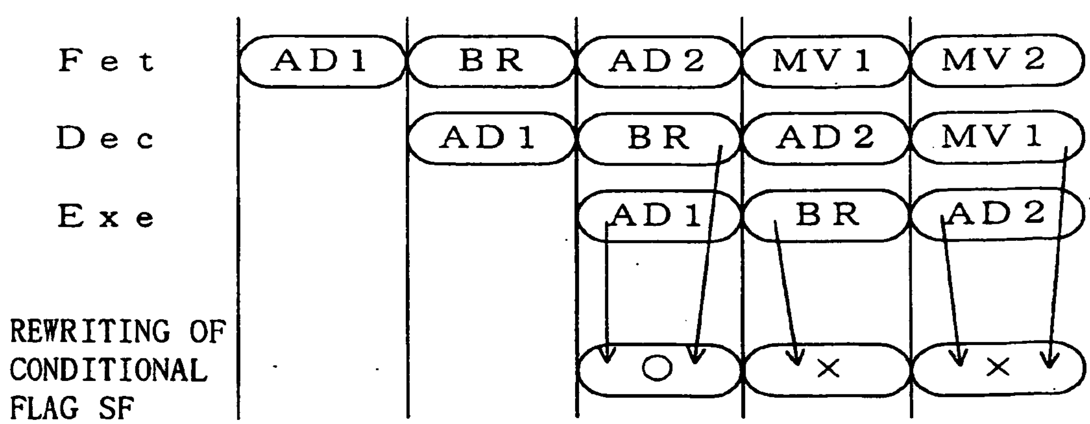

Fet
Dec
Exe
REWRITING OF
CONDITIONAL
FLAG SF
AD1
BR
AD2
MV1
MV2
AD1
BR
AD2
MV1
AD1
BR
AD2
○
×
×

Fig. 8

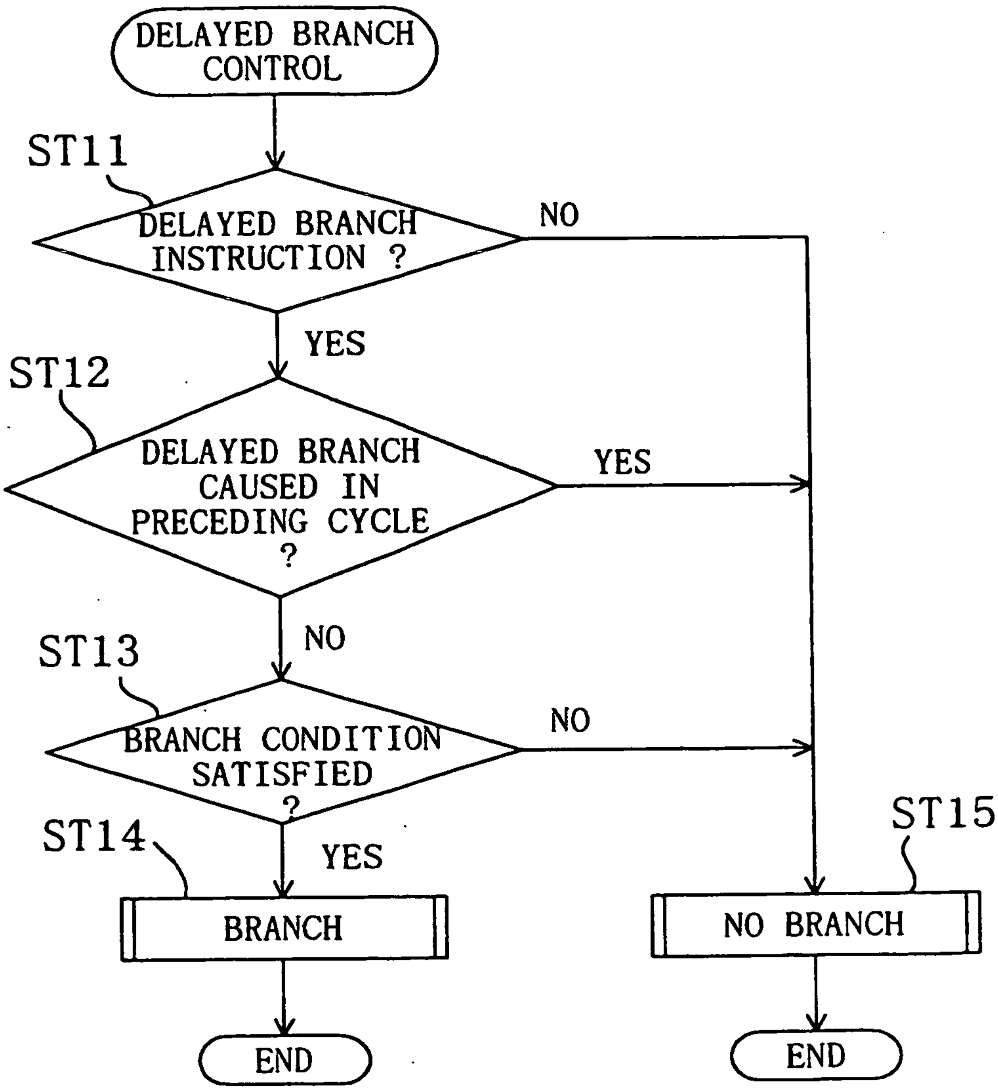

DELAYED BRANCH CONTROL
ST11
DELAYED BRANCH INSTRUCTION ?
NO
YES
ST12
DELAYED BRANCH CAUSED IN PRECEDING CYCLE ?
YES
NO
ST13
BRANCH CONDITION SATISFIED ?
NO
ST14
YES
ST15
BRANCH
NO BRANCH
END
END

Fig. 10

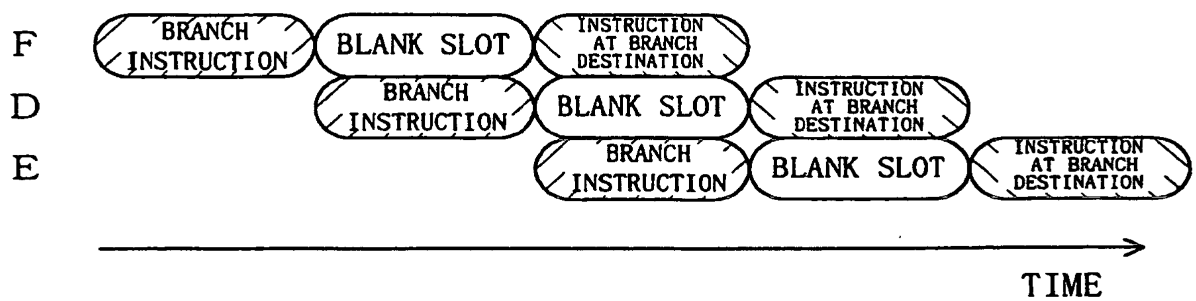


Fig. 11

ADDRESS	INSTRUCTION
100	br 200 (CONDITIONAL BRANCH TO 200)
101	br 400 (CONDITIONAL BRANCH TO 400)

Fig. 14

ADDRESS	INSTRUCTION	CONDITIONAL BRANCH			
100	br200	SATISFIED	SATISFIED	UN SATISFIED	UN SATISFIED
101	br400	SATISFIED	UN SATISFIED	SATISFIED	UN SATISFIED
ADDRESS TO BE EXECUTED		100 *** 200 201 202 203 ⋮	100 *** 200 201 202 203 ⋮	100 101 *** 400 401 402 ⋮	100 101 102 103 104 105 ⋮